Designing Resilient and Secure Smart Software Systems Engineering Foundations, AI-Assisted Development, and Runtime Assurance

Modern software systems increasingly rely on data-driven components, distributed architectures, and adaptive behaviors. While these technologies enable flexibility and scalability, they also challenge traditional software engineering assumptions such as determinism, static behavior, and centralized control. In this context, embedding intelligence without strong engineering discipline often results in fragile, unsafe, and difficult-to-maintain systems. This seminar presents an engineering-centric perspective on the design and operation of resilient and secure smart software systems. The central idea is that artificial intelligence must be treated as a controlled and engineered software component, embedded within well-defined architectures, lifecycle processes, and runtime supervision mechanisms, rather than as an autonomous replacement for software engineering.

The talk introduces a co-design loop between AI and software engineering, where AI assists engineers in structuring informal artifacts (requirements, logs, incident reports), generating candidate engineering solutions, and supporting system evolution and remediation. At the same time, software engineering principles—such as architectural separation of concerns, isolation, traceability, controlled deployment, and runtime feedback—ensure that smart components remain safe, resilient, and accountable throughout the system lifecycle. Through concrete examples and case studies in federated learning, cyber-physical systems, and digital twins, the seminar illustrates how resilience and security can be engineered both at design time and at runtime, enabling systems to tolerate faults, adversarial behavior, and unforeseen operating conditions without loss of trust. The seminar concludes with a research vision toward Trustworthy Software Engineering, where resilience, security, and adaptability are achieved through systematic engineering rather than post-hoc validation.